Зуйкевич Лидия, 7 группа

Лабораторная работа №4

Вариант 7

**Задача:**

Задача о рюкзаке (англ. Knapsack problem) — дано n предметов, предмет i имеет массу

и стоимость . Необходимо выбрать из этих предметов такой набор, чтобы суммарная масса не превосходила заданной величины W (вместимость рюкзака), а суммарная стоимость была максимальна.

Рассмотрим задачу Неограниченный рюкзак (англ. Unbounded Knapsack Problem), в которой любой предмет может быть выбран любое количество раз.

Формулировка задачи:

Каждый предмет может быть выбран любое число раз. Необходимо выбрать количество предметов каждого типа так, чтобы:

максимизировать общую стоимость: ;

выполнялось условие совместности: ;

где целое, для всех i = 1, 2, . . . , n.

**Условие:**

Это и есть математическая модель нашей задачи.

**Алгоритм решения:**

Для решения воспользуемся методом динамического программирования.

Пусть n - количество типов предметов, W - вместимость рюкзака.

У нас есть массивы (в скобках указаны размерности):

price[n] – цена единицы предмета каждого типа.

weight [n] – вес единицы предмета каждого типа.

Двумерные массивы, которые будем заполнять в ходе решения: f[n+1][W+1] и p[n][W+1]. Индексация всех массивов начинается с 0.

*Для начала заполним таблицу:*

Пусть f[i][j] - максимальная стоимость любого количества вещей типов от 1 до i, суммарным весом до j включительно.

Заполним f[0][j] f[i][0] нулями (логически ясно, что эти стоимости равны 0).

Тогда меняя i от 1 до n, рассчитаем на каждом шаге f[i][j] для j от 1 до W, по рекуррентной формуле:

f[i][j] =

p[i][j] =

После выполнения в f[n][W] будет лежать максимальная стоимость предметов, помещающихся в рюкзак.

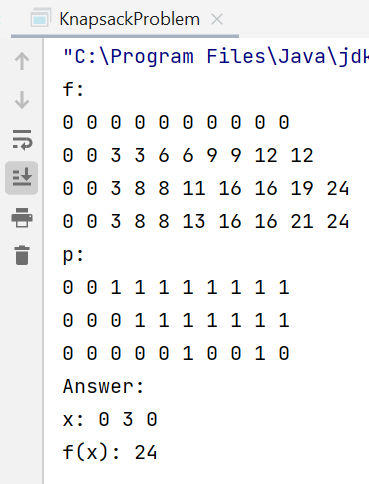
*Обратный ход:*

Массив нужен для обратного хода. Значение [i][j] означает, берем ли мы единицу предмета типа i в рюкзак вместимостью j. На этом основан алгоритм обратного хода: мы начинаем с вместимости W и предмета типа n, изначально количество предметов каждого типа . Если p[n][W] = 1, то x[n – 1] = x[n – 1] + 1, а оставшаяся вместительность рюкзака W – weight[n – 1], тогда переходим к элементу p[n][W - weight[n – 1]]. Если p[n][W - weight[n – 1]] = 0, то переходим к другому типу предмета для той же вместимости, т.е. к элементу p[n - 1][W - weight[n – 1]]. Если же изначально p[n][W] = 0, то также переходим к другому типу предмета для той же вместимости, т.е. к элементу p[n - 1][W].

**Листинг программы:**

public class KnapsackProblem {  
 private int n;  
 private int capacity;  
 private int[] x;  
 private int[] weight;  
 private int[] price;  
 private int[][] f;  
 private int[][] p;  
  
 KnapsackProblem(int n, int[] w, int[] p, int W){  
 this.n = n;  
 this.capacity = W;  
 this.weight = w;  
 this.price = p;  
 this.f = new int[n + 1][W + 1];  
 this.p = new int[n][W + 1];  
 this.x = new int[n];  
 }  
  
 public int[][] getF() {  
 return f;  
 }  
  
 public int[][] getP() {  
 return p;  
 }  
  
 public void fill\_table(){  
 for (int i = 0; i <= capacity; i++){  
 f[0][i] = 0;  
 }  
  
 for (int i = 1; i <= n; i++) {  
 f[i][0] = 0;  
 }  
  
 for (int i = 1; i <= n; i++) {  
 for (int j = 1; j <= capacity; j++) {  
 if (j >= weight[i - 1]) {  
 f[i][j] = Math.*max*(f[i - 1][j], f[i][j - weight[i - 1]] + price[i - 1]);  
 }  
 else{  
 f[i][j] = f[i - 1][j];  
 }  
 p[i - 1][j] = f[i][j] == f[i - 1][j] ? 0 : 1;  
 }  
 }  
 }  
  
 public void find\_answer(){  
 int W = capacity;  
 int i = n - 1;  
  
 while (W > 0 && i >= 0){  
 while (p[i][W] != 0 && W != 0){  
 x[i] += 1;  
 W -= weight[i];  
 }  
  
 if (W == 0){  
 return;  
 }  
  
 i--;  
 }  
 }  
  
 public void print\_data(int[][] arr){  
 for (int[] row : arr){  
 for (int elem : row) {  
 System.*out*.print(elem + " ");  
 }  
 System.*out*.println();  
 }  
 }  
  
 public void print\_answer(){  
 System.*out*.println("Answer: ");  
 System.*out*.print("x: ");  
 for (int elem : x){  
 System.*out*.print(elem + " ");  
 }  
 System.*out*.println();  
 System.*out*.println("f(x): " + f[n][capacity]);  
 }  
  
 public static void main(String[] args){  
 int[] p = new int[] {3, 8, 13};  
 int[] w = new int[] {2, 3, 5};  
 KnapsackProblem obj = new KnapsackProblem(3, w, p, 9);  
 obj.fill\_table();  
 System.*out*.println("f: ");  
 obj.print\_data(obj.getF());  
 System.*out*.println("p: ");  
 obj.print\_data(obj.getP());  
 obj.find\_answer();  
 obj.print\_answer();  
  
 }  
}

**Вывод программы:**



**Анализ результатов:**

Таблица в таком виде, как в примере:

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| w |  |  |  |  |  |  |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 1 | 0 | 0 | 0 | 0 | 0 | 0 |
| 2 | 3 | 1 | 3 | 0 | 3 | 0 |
| 3 | 3 | 1 | 8 | 1 | 8 | 0 |
| 4 | 6 | 1 | 8 | 1 | 8 | 0 |
| 5 | 6 | 1 | 11 | 1 | 13 | 1 |
| 6 | 9 | 1 | 16 | 1 | 16 | 0 |
| 7 | 9 | 1 | 16 | 1 | 16 | 0 |
| 8 | 12 | 1 | 19 | 1 | 21 | 1 |
| 9 | 12 | 1 | 24 | 1 | 24 | 0 |

Ответ: x = {0, 3, 0}, f(x) = 24, т.е. берем 3 предмета типа 2, и получаем максимальную стоимость 24.